Real-Time Procedural Terrain Generation with Marching Cubes

Joseph Chambers-Graham

Contents

1 Introduction

- project and report overview

2 Background

2.1 Signed Distance Functions

- what is a SDF
- operations on SDF
- conventions (what sign is in/out?)
- aside: “density” functions? An nvidia paper says this, but everyone else says distance, which makes more sense

2.1.1 Noise

- brief explanation of noise for terrain generation, how to fit it into a SDF

2.1.2 Approximation

- not all of the SDFs are exact distances - particularly noise - but good enough for rendering
- is it good enough for physics?

2.2 Marching Cubes Algorithm

- explain algorithm - basically just a reference?

3 Design

3.1 GPU Implementation

- How is the algorithm modified
- speed comparison (GPU is much faster)

3.2 LOD system

- cannot render everything at max resolution

3.2.1 Octree

- Chunks with same point count, but scaled based on depth of octree

3.3 Transvoxel Algorithm

3.3.1 The problem it solves

- cracks in between chunks at different LOD

3.3.2 How the algorithm works

- basically just a reference again
- reference implementation is complicated so could do with explanation

3.3.3 Adaption of the algorithm to GPU

- optimizations to make use of sequential generation have actually been removed
- regardless, this is still pretty fast on the GPU
- similar approach to parallel marching cubes

3.3.4 More issues

- zero width triangles
- edge cases where it still goes wrong (maybe a way to fix this by revisiting the octree)

3.4 Terrain Modification

3.4.1 Strategy

- adding or removing primitives using SDF operations
- regenerating chunks
- when to regenerate chunks, and which ones
- how many primitives is reasonable before slowdown due to SDF evaluation?
- bounding boxes of primitives to increase this limit - dont need to evaluate parts not affected

3.5 Physics

- Just use bullet physics - well-known and optimized library
- zero-width triangles havent turned out to be a problem so far
- generating physics meshes is slow - need to find a solution to this

3.5.1 SDF-based physics

- perhaps look at this - might be a struggle because of non-exact SDFs, and how would we do it for anything that isn't a sphere anyway?

4 Implementation

- C++ with fairly low-level opengl libraries
- key code probably in design section already
- section for the "boring bits" of the implementation, rather than key ideas
- perhaps section to explain other things that needed bugfixes?
- something tying it all together?

5 Conclusion

- demonstration of final product
- conclusion
- somehting about future work?